VARIABLE MASKING FOR SEGMENTED MEMORY

Patent Number: US3800292
Publication date: 1974-03-26
Inventor(s): CURLEY J; MARTLAND W
Applicant(s): HONEYWELL INF SYSTEMS
Requested Patent: DE2350225
Application Number: US19720295303 19721005
Priority Number(s): US19720295303 19721005
IPC Classification: G06F3/00
EC Classification: G06F12/04; G06F12/06; G06F12/08B10; G06F13/18; G11C29/00R
Equivalents: CA1002204, FR2202611, GB1433393, JP1120884C, JP49074447, JP57010498B

Abstract

A variable masking technique and apparatus for simultaneously accessing under program control a variable number of bytes b of information stored in groups of i bytes per group in any of n modules of a storage device that is capable of operating in any of m storage modes. Storage is provided by a multi-level system providing multiple levels of storage comprising a high speed low capacity storage device (buffer store) coupled serially to successive levels of lower speed, high capacity storage devices including means for varying the number of bytes to be simultaneously accessed from any of the storage devices.

Description

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to computer multi-level storage systems and more particularly to storage hierarchies having a high speed low capacity storage device coupled to successive levels of lower speed, high capacity storage devices of n-modules, and including means for varying the number of bytes that are simultaneously accessed from any of the n-modules of said high capacity storage devices.

2. Description of the Prior Art

The storage hierarchy concept is based upon the observed phenomenon that individual stored programs, under execution, exhibit the behavior that in a given period of time a localized area of memory receives a very high frequency of usage. Thus a memory organization that provides a relatively small size high-speed buffer at the central processing unit (CPU) interface and the various levels of increasing capacity slower storage can provide an effective access time that lies somewhere in between the range of the fastest and the slowest elements of the hierarchy and provides a large capacity memory system that is "transparent" to the software.

To date all noteworthy storage level implementations of the invisible storage hierarchy storage system have consisted of the IBM 360/85, 370/155 and 370/165 which consist of two levels of storage, the first level of storage consisting of a high speed solid state buffer termed a "cache memory, " high speed associative logic techniques and high speed control logic to control the fully interleaved two by four by eight way, second level store. The second level store in the 360 system is bulk core storage and in the 370 systems can be either bulk core or metal oxide semiconductor integrated chips (MOSIC). A general description of the system/ 370 model 165 (cache memory) can be found on pages 214-220 of a book by Harry Katzen, Jr. entitled "Computer Organization and the System 370 " and published in 1971 by Van Nostrand Reinhold Company. The IBM 360/85 is described generally on pages 2-30 of IBM System Journal, Volume 7, No. 1, 1968.

Some mapping schemes for buffer store can be found in an article by C.J. Conti on storage hierarchies entitled "Concepts for Buffer Storage" and published in Computer Group News, March 1969, pages 10-13. Briefly a sector mapping scheme is described which requires large scale associative techniques of large scale integrated content-addressable memories (LSICAM) implementation or discrete logic type implementation; this technique is utilized in some of the 360 systems. Two and four level set associative algorithm techniques for buffer store mapping is utilized in

the 370/155, 165; these techniques are also described in the above mentioned Conti article and may be implemented by a two or four level ranked comparator implementation. Memory block replacement in all cases is of the least recently used (LRU) block type, whereas a least frequently used (LFU), a working set, and a first in-first out (FIFO) arrangement may be utilized for replacement algorithms.

In prior art buffer store systems of which the Applicants are aware the buffer store performs local and store operations in one mode upon command from the central processing unit (CPU). Whenever a CPU performs a load operation and the addressed information resides in the buffer store, the buffer store presents the information to the CPU at buffer memory high speed. If the addressed information does not reside in buffer store, control circuitry in the buffer store effects a transfer of a block of information from main store (MS) to buffer store and gives the CPU the requested information from this block. For CPU stores operations, the information is sent from the CPU to MS. If the addressed location for this store operation is in the buffer, then that buffer store location is also updated.

It is sometimes desirable to completely by-pass buffer store when for some reason or another it becomes inoperable; or it is sometimes desirable to reduce the buffer memory size where the customer's needs permit lower performance in order to effect lower cost. Moreover in solving certain problems the full "cache" maping technique is not necessary and a full block load need not be loaded into buffer store subsequent to each read miss. Moreover it is desirable to vary the number of bytes that may be simultaneously said concurrently accessed from any of the n-modules of the high capacity storage devices.

OBJECTS

It is an object, therefore, of the invention to provide an improved multi-level storage system.

It is another object of the invention to provide a device having a multi-level storage system capable of multi-mode mapping of buffer store, and wherein a variable number of bytes may be simultaneously accessed at any one time.

It is still another object of the invention to provide a device having a multi-level storage system capable of dynamically bypassing buffer store, and wherein the number of bytes that may be simultaneously accessed at any one time may be varied.

Yet another object of the invention is to provide a device having a multi-level storage system wherein the buffer store capacity is variable, and accessing of information is also variable.

Other objects and advantages of the invention will become apparent from the following description of the preferred embodiment of the invention when read in conjunction with the drawings contained herewith.

SUMMARY OF THE INVENTION

The foregoing objects are achieved, according to one embodiment of the instant invention by providing for multiple levels of storage comprising a high speed low capacity buffer store coupled serially to successive levels of lower speed, high capacity devices, and including means for varying the number of bytes to be simultaneously accessed from any or all of the storage devices.

A buffer store module normally is arranged in two modules of 128 columns each, with each column capable of storing one block of information comprising 32 bytes per block. The buffer store has means for operation in normal mode generally referred to as 128 .times. 2 .times. 32, i.e. two modules of 128 columns each storing one block per column. Another mode of operation is the 128 .times. 2 .times. 16 wherein the buffer store has two modules of 128 columns each storing one/half a block, i.e., 16 bytes, per column. Another mode of operation is the 256 .times. 2 .times. 16 mode wherein the buffer store has two modules of 256 columns, each column containing half a block of information, 16 bytes. The normal mode loads and accesses the backing store modules for either 16 or 32 bytes; thus giving a micro programmer greater flexibility for individual instruction performance optimization in micro programming. A Non-Allocate Mode - 8 byte fetch when four byte-groups are temporarily stored in Cache in a mode which forces all Cache references to "miss." Finally a mode is provided so that the buffer store may be completely bypassed. Means are also provided to mark any or all of M bytes in any or all of n-modules of said high capacity storage devices.

BRIEF DESCRIPTION OF THE DRAWINGS

This invention will be described with reference to the accompanying drawings wherein:

FIG. 1 is a block diagram of an overall view of the invention in its environment illustrating the multi-level storage system and controls thereof.

FIGS. 2A and 2B are block diagrams illustrating address arrangements utilized by the invention.

FIG. 3 is a more detailed block diagram of the major components of the invention within their environment.

FIGS. 4, 5, 6 and 7 are detailed logic block diagrams illustrating features of the invention.

FIGS. 8a through 8d are logic block diagrams of the masking and mode selection structure of the invention.

FIG. 8e is a logic block diagram of mode selection of the invention.

FIG. 9a shows timing diagrams of the invention.

FIG. 10 is a prior art schematic diagram showing the inventions for signals and symbols utilized FIGS. 8A-8E.

DESCRIPTION OF A PREFERRED EMBODIMENT

General

Referring to FIG. 1 there is shown in diagram format a multi-level storage system providing for multiple levels of storage comprised herein of the buffer store 104 and the main (back-up) store 101. The buffer store memory 104 is typically a semiconductor bipolar random access memory array of 8,192 bytes. The cycle time of the buffer memory is typically 150 nanoseconds having a typical access time of 95 nanoseconds. The main store 101 is normally a four-way interleaved random access memory comprised of four MOS memory modules 101A-D. Main store is typically organized so that 32 consecutive bytes are spread over the four storage units 101 i.e. location 0 is in storage unit 101A; location 8 is in storage unit 101 B, etc. Cycle time of the main memory 101 is typically 0.8 microseconds. It can be readily observed that the buffer store 104 is a high speed memory which is several times faster than the main memory (back-up) store.

A buffer store directory 105 is utilized to store row-addresses of the data that is stored in buffer store 104. The buffer store directory 105 comprises typically an array of 128 .times. 36 bits and has a cycle time of 150 nanoseconds with an access time of 75 nanoseconds. The buffer store 104 has as its main function the storage of the contents of those parts of main store 101 currently being used by the processor; therefore the processor can fetch a great majority of the information it needs by accessing the high speed buffer store memory 104. When the program shifts its operations from those requiring the information from that portion of main memory currently in buffer store memory to those operations requiring information currently residing in another portion of main memory, then that portion of main memory is loaded into the buffer store memory. The main store sequencer 102 (which is the subject of another invention invented by others at Honeywell Information Systems Inc. and is the subject of another application) provides the interface between the main store 101 and the buffer store control 103. The buffer store control, although shown as a box, may not necessarily be centrally located, and typically includes such logic circuitry as shown on FIGS. 8A-8E at paths 106, 107, 108 and 109 between the modules of the main store and between the main store 101 and the main store sequencer 102 is 8 bytes wide which may change to 16 bytes; moreover data paths 114, and 115 between the main store sequencer 102 and buffer store central 103 between buffer store control 103 and buffer store memory 104, and between the main store sequencer 102 and the input/output control unit 10C, (not shown) are 8 bytes wide. Data paths 110 from the central processing unit CPU (not shown) and the buffer store control unit are also typically 8 bytes wide; however data path 113 from the buffer store control unit to the CPU is four bytes wide.

Because individual stored programs in back-up store (in this instance main store 101) which are under execution at a given time are generally to be found in a localized area or in areas dispersed throughout the available memory of main memory 101; that area is placed in buffer store memory 104 during current program execution and by accessing the currently required information from buffer store memory 102, the effective main storage access time is significantly reduced.

The input/output control unit IOC (not shown) does not directly reference the buffer store memory 104, but rather it communicates with main store 101 via main store sequencer 102; consequently the buffer store 104 is purged whenever store operations are made into memory locations currently being executed and contained by the buffer store 104.

In the storage hierarchal system of FIG. 1, only two levels are shown, buffer store 104 and main store 101, although many other levels may be used. Generally the highest level store is termed the local store, sometimes also known as the "cache" memory, whereas the lowest level store is known as the backing store. The highest level store has generally the fastest access time but also generally has the smallest storage capacity. In FIG. 1, since there are only two levels of storage the "cache" corresponds to buffer store memory 104 and the back-up store corresponds to main store 101. Each storage device in the hierarchy is partitioned logically into blocks bn, each block being comprised of 32 bytes. The buffer store in normal mode is typically organized into two 128 column modules (see later discussion). Each column of buffer store may contain one block of information consisting of 32 bytes. The main store 101 may contain many blocks bn of 32 byte information arranged in columns and rows.

Referring now to FIG. 2A there is shown a block diagram of an address structure 200 utilized to address the buffer store memory 104. The structure of FIG. 2A is that part of an instruction, that identifies an address space in the buffer

store 104 and relates that buffer address to an address in main store 101. The address structure 200 is typically 24 bits in length. It begins with bit 8, because prior bits are not pertinent to the address. Address field 201 comprises bits 8 through 10 – a total of 3 bits. Address field 201 is a reversed address space to provide additional addressing capacity for addressing from an expanded main store. Row address field 202 consists typically of 11 bits through 19 – a total of 9 bits; whereas column address field 203 consists typically of bits 20 through 26 – a total of 6 bits. Double word address field 204 consists typically of two bits numbered 27 and 28; word address field 205 consists typically of one bit numbered 29; and byte address field 206 consists typically of two bits 30 and 31. (The functions of these address fields will be described infra.)

Referring now to FIG. 2B there is shown a typical structure of an address space 250 typically contained in a portion of buffer store directory 105. The address space 250 is typically 36 bits in length and typically comprises a four bit parity field 251, a two bit buffer count field 252, four validity one bit fields 253 - 256, a 12 bit row lower field, a 12 bit row upper field, a one bit activity field 259, and a one bit OK field 260. Column field 203 (FIG. 2A) is used to address buffer store directory 105; by utilizing bits 27 and 28 together with column field 203 the buffer store 104 may also be addressed; row field 202 of address space 200 is used for comparison to row lower field 257 and row upper field 258 which are resident in buffer store directory 105. A successful comparison is herein termed a "hit" and indicates that the required information of main memory resident at the row field 202 of address space 200 is also resident in buffer store and is located in a column of buffer store 104 designated by column field 203. The parity field 251 is utilized to ascertain the correctness of information contained in the address space 250. A parity bit is formed on the following bit fields: buffer count field 252, valid bit fields 253, 254, 255, and 256, and OK field 260. When reading a directory word, parity is checked against these bits. On the remaining 24 bits the three parity bits are checked when reading, and regenerated when writing into the directory. The buffer count field 252 stores possible error occurrences with respect to a particular buffer store directory location. Three error occurrences are stored and permitted and on the fourth error occurrence that particular location in the buffer store directory to which reference is made is invalidated. Validity bits 253 and 255 point to row upper location while validity bits 254 and 256 point to row lower locations, and are utilized to indicate the validity of data contained in the referenced location. For example, when a "hit" (successful compare) is made in buffer store directory, the validity bits for that location are also examined; if a logical 1 is present the data in buffer store is valid and may be utilized, but if a logical 0 is present it indicates that the data in buffer store is not valid or representative of the comparable data in main store because of possible alteration of that main store location by an input/output (I/O) unit or because of other errors or it has never been loaded. The activity field 259 indicates the least recently used upper or lower rows in the buffer store directory and is utilized as part of the algorithm that selects a location to write in new data when a "no hit" (unsuccessful compare) occurs. The OK bit 260 indicates that the word associated with it has no errors, i.e., the word 250 has not been invalidated by an error field. A logical 1 indicates the error count has not been exceeded; a logical 0 indicates errors.

Referring now to FIGS. 3 and 4, the Central Processing Unit CPU 306 issues an address comprising bits 8-29 of FIG. 2A together with a command for action by the buffer store system 300. The issued address is stored in memory address unit 307 which contains storage flip-flops, decode logic appurtenant logic circuitry (not shown) and generates signals, by means known in the art, for addressing generally the data upper module 304U, data lower module 304L, and the buffer directory module 305. (The data upper and lower modules 304U and 304L are more detailed views of buffer store memory 104 of FIG. 1.) Bits 20-26 of FIG. 2A are utilized to address the buffer directory module 305, bits 20-29 are utilized to address the data buffer modules 304U and 304L, (note the reuse of bits 20-26 for this purpose) and bits 8-19 are utilized for comparison via compare unit 308 to information stored in buffer directory module 305. Referring to FIG. 4 the data upper and lower modules 304U and 304L are further subdivided to upper and lower banks 401, 402 and 403, 404 respectively; whereas buffer directory module 305 is further subdivided into row upper fields 405 and row lower fields 406. Each of the data in row upper and lower fields 405 and 406 which comprise information arranged in row upper and lower fields 258 and 257 respectively in accordance with word type 250 of FIG. 2B, are compared in comparator 308 to the data contained in the row address field 202 of word type 200 issued by the CPU 306. If a successful compare "hit" results, it may be a hit upper or a hit lower, indicating that the successful compare was with row upper 405 or row lower 406 respectively of buffer directory module 305 and that the information desired is in buffer store in the data upper module or data lower module depending on which row (upper or lower) of the buffer directory the "hit" occurred. (Note that a hit in "row upper or row lower" of the buffer store directory indicates the information is in either the upper or lower module 304U or 304L respectively but does not indicate the row (i.e., bank upper or bank lower) within the upper or lower module. When a hit occurs one word comprising 8 bytes of data may be read out into selector 309 from any one of the data module banks. However it will be noted from prior description that while data from the CPU to the buffer store is over an 8 byte path (used generally for write operations into buffer), data from the data buffer store to the CPU is transmitted over a path only 4 bytes wide (used typically in reading from buffer and supplying information to CPU). Moreover it will be noted from FIG. 4 that each upper and lower module 304U and 304L respectively are further organized into 128 columns Cn, each column capable of holding one block of information i.e. 32 bytes. Each upper and lower module 304U and 304L respectively is further subdivided into upper and lower banks (i.e., rows of the upper or lower module) 401, 402, 403, and 404 respectively, having the same 128 columns as the data modules 304U and 304L, but each column of each bank contains two words, i.e., 16 bytes; hence each bank (i.e. row of each buffer store module) contains 2,048 bytes, with each data module containing 4,096 bytes, and with the entire buffer store memory 108 containing 8,192 bytes.

Assuming, for example, that a hit upper occurs in the directory 305 referencing word 511 in upper bank 304U, and the CPU has requested a read operation, i.e., desires 4 bytes that currently reside at the addressed location, and moreover

desires the first 4 bytes of word 511 located in upper bank 401 of upper data module 304U. (If a full 8 bytes were needed as in write operations, bits 27, 28 would be utilized thus addressing the entire upper module 304U.) In this example, address bit 29 of FIG. 2A is not set, i.e., is represented by a logical 0; hence a low signal representing address bit 29 and AND gate 407 provides an enabling signal on one of the terminals of AND gate 407 and a disabling signal on one terminal of AND gate 408. Hence with the upper banks of upper and lower modules 304U and 304L respectively, selected, and with address bit 29 not set therefore referencing 4 bytes on the same column of two different modules, i.e., words 511 and 511b, a conflict results since at this juncture there is no way of knowing whether or not to deliver 4 bytes from the upper bank of the upper module or the lower module. The conflict is resolved by AND gates 410 and 411 respectively which has an enabling signal on one or the other of the gates depending upon which module --upper or lower-- is referenced by the hit in directory 305. In this instance AND gate 410 is enabled, since the hit referenced the upper module, and the first four bytes of word 511 are selected. Note that logic circuitry 490 is the upper bank selection circuitry of upper and lower modules, 304U and 304L whereas logic circuitry 491, only a part of which is shown since it is similar to logic circuitry, 490, is lower bank selection circuitry for upper and lower modules 304U and 304L. The next 4 bytes are selected by initiating a new operation by the CPU wherein the address is the same except address bit 29 which is the 1's complement of its state during the previous operation. When a write operation is requested an 8 byte word is required and this is selected by circuitry to be later described utilizing 27, and 28 of double word field 204.

When a no hit condition is encountered the data requested by the CPU is not in the buffer store and most be retrieved from main memory 301. Since main memory 301 is comprised of four modules 301A-301D, and since a block of information is normally four-way interleaved with 8 bytes in each of the main memory modules, each of these modules must be accessed in order to retrieve a block of information. During the first access from one of the main memory modules 301A-301D, 8 bytes of data are obtained and loaded into the buffer store at an address selected by the CPU through data switch 315; also 4 bytes of data are delivered to the CPU through data switches 315 and 311 respectively. The address is the incremented and another main memory request is made and another 8 bytes of data are loaded into the buffer store but 4 bytes more are not delivered to the CPU as in the previous cycle; this procedure is repeated two more times (a total of four accesses) until one block of information has been written into buffer store and a word lone-eighth block) of information has been delivered to the CPU. To obtain the remaining information the CPU will continue to address buffer store but because an entire block of information has been delivered to buffer store, a hit will result and the information will then be delivered from buffer store without making further access to main memory 301 (assuming that it has not been purged by the I/O). The CPU addresses the buffer directory 305 through I/O address and control unit 312 and 2 .times. 1 switch 310. The 2 .times. 1 switch 310 permits the use of two addresses, one for the main memory 301 and the other for the buffer directory 305 with only one address being directed to the buffer directory of main memory.

Referring again to FIG. 3, CPU - 306 addresses the buffer directory module 305, via memory address unit 307. Memory address unit 307 is also utilized to address the address control 350 and the 2 .times. 1 switch 310. When the CPU directs that data be written into the buffer store or into the main memory modules data write switch 315 is utilized to select the proper unit. The CPU - 306 may desire data from either the buffer store having data modules 304U, 304L, or from main memory 301 and the selection is accomplished by a data read switch 311. Sometimes it is necessary that the IOC unit 307 address buffer store I/O address control unit 312; this is accomplished by a 2 .times. 1 switch 310 which determines whether the CPU-306 or IOC-307 will be permitted to adjust the buffer directory module. If there is a conflict is is resolved through the priority resolution unit 351 in cooperation with the buffer control unit 303. See co-pending patent applications Ser. Nos. 295,331 and 295,417.

The main storage sequencer (MSS) generally denoted as 300A is the subject of another invention as hereinbefore mentioned and is included herewith for completeness and as background for the instant invention. See co-pending patent application Ser. No. 295,331. An MSS control 352 is utilized to determine whether or not main memory is busy and to store and issue signal acknowledging request to main memory and providing information as to the current status of main memory. It also typically communicates with priority resolution unit 351, address control 350, and data read switch 311. Reconfiguration unit 353 receives signals from the CPU and according to their request maps main memory 301 into various modes via main memory module switch 354 which may typically be nothing more than a multiplexor. See co-pending patent application Ser. No. 295,417. Address control unit 350 is under MSS control and is utilized to gate the I/O, CPU, or buffer store addresses, to the main memory 301.

Referring now to FIG. 5 there is shown a second mode of operation of the buffer store memory system 300. When a user can trade off some speed and capacity in order to realize some economic benefits the mode sometimes called 128 .times. 2 .times. 16 is utilized. In this mode of operation there is half the buffer memory size of the previously described normal mode. For ease of understanding FIG. 5 has been arranged similar to FIG. 4; however, it will be noted that no lower banks exist in upper and lower modules 504U and 504L respectively. Hence there is 2,048 bytes in upper bank 501 and 2,048 bytes in upper bank 503 resulting in a total of 4,096 bytes for the buffer memory 104. The terminology, again for convenience, of buffer store directory 505D has been left similar to the terminology of buffer store directory 305 of FIG. 4 since both make reference in accordance to fields 257 and 258 of address space 250 contained in buffer store directory rather than making reference to the buffer store memory 104. The information in row upper 505 and row lower 506 of buffer store directory 505D, however, do make reference to buffer store memory 104 and is utilized as previously described. It will be noted by further examining upper banks 504U and 504L respectively that there are 128 columns in both upper banks but each column is now capable of storing only a half a block or 16

bytes since the populated boards 502 and 504 are not utilized. The operation of this mode is similar to the normal mode previously described, however, there are only two accesses to either the upper or lower module because only a half a block of information need be read or written into cache in any one column of any one module. The word selection circuitry 590 of FIG. 5 is also different from the word selection circuitry 490 and 491 of FIG. 4 since only half the circuitry is needed to select the reference upper bank in either the upper or lower module. The mode of FIG. 5 is fixed at the factory and provides faster speeds since only 16 bytes need only be accessed in any column thus requiring half the number of accesses by the buffer.

The mode of operation depicted in FIG. 6 is known as the 256 .times. 3 .times. 16 mode. Referring to FIG. 6 the upper and lower modules 604U and 604L are each arranged in 256 columns, each column capable of storing one 8 byte word. In other words each bank 601, 602 of upper module 604U has a capacity of 2,048 bytes with each bank being 128 columns wide. The two banks, although shown in vertical relation one to the other in order to relate more easily to the other modes, are actually better pictured as arranged continuously from column 1 to column 256 with 8 byte words 1 and 2 in column 1 and 8 byte words 1023 and 1024 in column 256. The lower module 604L may be similarly pictured. The directory 605D in this mode utilizes the entire memory space allotted to it whereas in previous modes it will be noted that only half the memories space allotted to it was utilized. The remaining elements such as the logic selection circuitry 690 and 691 is similar to that of FIG. 4. On a hit condition utilizing this mode of appropriately referenced column 1 through 256 is accessed 4 bytes of data is given to the CPU in the read mode. On a no hit condition main memory is accessed only twice and each time 8 bytes of data is loaded into the buffer store memory with 4 bytes being delivered to the CPU during the first MS access. Whereas this mode, the 256 .times. 2 .times. 16 mode, arrogates to itself the advantages of the 128 .times. 2 .times. 16 mode and eliminates the capacity disadvantage, it is nonetheless sometimes desirable to have the capability of loading or delivering from any referenced column either a full block or a half a block depending upon the requirements of the programmer. The mode of FIG. 7 the 128 .times. 2 .times. 32/16 mode is capable of performing in this manner.

Referring to FIG. 7 the upper module 704U has an upper and lower bank 701, 702, however, each upper and lower bank is further subdivided in capacity resulting in two one half upper banks each having a capacity of one half the full bank. This division is effected in all banks of all modules. The remaining elements of FIG. 7 the selection circuitry 790 and 791 and the directory 750D are similar to the normal mode of FIG. 4. Thus the micro programmer has the modes of FIGS. 4, 6, and 7 to manipulate as the requirements of the micro program dictate. The mode of FIG. 5 as previously noted is predetermined and fixed at the time the system is acquired; however, it may be converted to the modes of FIGS. 4, 6, and 7 by including the required additional lower banks and the selection circuitry therefor.

Referring now to FIG. 10 there is shown a prior art diagram of various circuits in order to illustrate the conventions utilized herein. In order to simplify the multitude of complex logic circuits required in a design of a specific computer and to automate the preparation and reading of such design plans once the design has been approved, PLEXEDIT listings of logic functions (i.e. logic signals) are utilized. From such PLEXEDIT listings detailed logic block diagrams such as shown on FIGS. 8A through 8E may be prepared, or logic block diagrams once designed, PLEXEDITS may be prepared. The technique for reading PLEXEDIT listings and utilizing them is described in book 3 of a book entitled "Computer Fundamentals," copyrighted 1969 by Honeywell Inc. FIG. 10 does not represent any specific circuit of the invention but a description of it and the conventions utilized will enable the person of ordinary skill in the art to read FIGS. 8A through 8E and practice the invention.

A signal BXXXXXX is applied at input terminal 1000. The signal has been given the name BXXXXXX where B and 1 or X may be any letter or numeral; generally the first two characters in this case BX specify a major and minor logic area or a major logic area and a logic function. In this instance, B indicates the major logic area belonging to the buffer store. The third, fourth and fifth X's are reserved to specify the function (i.e. logical signal), and this function name may be varied according to the needs of the designer. The next to the last character, in this particular instance the sixth position, provides information as to the state of the signal i.e., whether or not it is an assertion or negation. For example, when the signal BXXXXXX passes through AND gate 1001 and through amplifier 1002 there is a first assertion. This first assertion is indicated by the next to the last character which in this case is a 1 (assertions are indicated by an odd number of the next to the last character and negations are indicated by an even number of the next to the last character). Following the BXXXXXX through AND gate 1003 and through another amplifier 1004 there is a second assertion indicated by the next to the last character which is a 3; as the signal continues and divides first through AND gate 1005 and then through amplifier 1006 there is another assertion indicated by the number 5 in the signal named BXXXX50 which indicates this is the third assertion of the signal. From the output of amplifier 1004 it is noted that the signal also divides and passes through AND gate 1009 and then through amplifier 1010 which again is the third assertion but now it is at a second level of the circuit and that level, in this case, is a 1; had there been a third level also the last character would have been a 2 and so on. Now the original signal BXXXXXX which is applied to input terminal 1000 is also applied to AND gate 1011 and Inverter 1012 producing a first inversion of the signal with this name and now shown as BXXXX00; the next to the last character is now a 0 indicating a first negation. As the signal continues through AND gate 1013 and inverter 1014 a second negation arises which is identified by the second to the last character being a 2 in the signal name BXXXX20.

Some further conventions shown on FIG. 10 and utilized in this disclosure follow. A filled in circle 1018 represents an internal source whereas a square such as 1019 represents an output connection pin. A small circle 1000 indicates an input connecting pin (except on the end of an amplifier, in which case it indicates an inverter). A square 1020

connected as shown on FIG. 10 indicates a flip-flop having output terminals 1021, 1022 to indicate the state of the flip-flop depending on which one is high. AND gate 1015 has two input terminals whereas the other AND gates shown have one input terminal. (Generally AND gates have more than one input terminal; however the single input AND gates are utilized herein to indicate that the signal is located similarly to a double input AND gate).

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to FIG. 8E there is shown as an example a partial logic block diagram for dynamically selecting the mode of operation of the invention under program control. (Similar logic block diagrams may be utilized for selecting any mode desired). More specifically, there is shown memory circuit 812E which comprises one module of the buffer store memory. AND gates 801E and 802E are OR'ed together to the input terminal of amplifier 803E whose output terminal is coupled to memory circuit 812E. This portion of the input circuit to memory circuit 812E utilizes bits 22 through 26 (see FIG. 2A) to address the appropriate column of the memory circuit 812E. The appropriate address shown as input bits (22-26) is applied to AND gates 801E and 802E. Whether or not memory circuit 812E is addressed by the CPU unit or I/O unit is determined by the input signal CPAGAT and I/0 AGAT which may be applied to AND gates 801E and 802E respectively. When the CPAGAT signal is high and the proper address is presented to AND gate 801E; it indicates that the CP is addressing the memory module 812E. Similarly, if the signal I/0 AGAT is high with the appropriate address applied to AND gate 802E it indicates that the I/0 unit is addressing the memory module 812E. Conflicts between the CP and the I/0 are resolved by priority resolution unit 351 of FIG. 3, which is the subject of an invention in application Ser. No. 295,331, filed on the same date as the instant application and assigned to the same assignee as the instant invention.

Once the appropriate column is selected it has previously been shown in connection with FIGS. 4, 5, 6 and 7 whether the word is in the upper or lower bank. How many bytes are delivered to or abstracted from buffer store depends also on the mode of operation previously described. FIG. 8E shows how this mode selection may be made. For example, if the 128 .times. 2 .times. =mode is desired wherein a 32 byte load is to be loaded or abstracted from buffer store, a function identified as B823210 is high; when other appropriate signals are also high on the same AND gate the mode of operation will be 128 .times. 2 .times. 32. When it is desired to operate in the 128 .times. 2 .times. 16 mode a signal identified by the name B821610 must be high. (See Table 1). Referring to FIG. 8E it will be noted that AND gates 804E and 806E are the CP and I/0 addressing gates for the 128 .times. 2 .times. 32 modes, i.e., when signal B823210 (the 128 .times. 2 .times. 32 mode signal) gate is high and signals CPAGAT and CPA20 (bit 20 on FIG. 2A) are also high, and AND gate 804E is enabled and the CP has access to the buffer store for a single 16 byte word. (It will be noted by referring to FIG. 2A that bit 27 of block 204 denotes a double word (32 bytes) whereas bit 20 of block 203 denotes a single word (4 bytes). If on the other hand the input signals on AND gate 806E are all high that is the signals I/0 AGT, (I/0 enabling signal) I/0 20 (bit 20), and B823210 (128 .times. 2 .times. 32 mode) are high, then AND gate 806E is enabled and the I/0 unit has access to the buffer store at the appropriate address previously addressed (as described supra) for a single word. By utilizing this analysis the other modes of operation may be also determined, since the physical and logic circuitry is similar in the lower buffer store module.

Referring now to FIGS. 8A through 8D, Exhibit I through VI and Table I (infra), there is shown logic block diagrams for mask control that controls the reading or writing of data in the appropriate row (i.e., upper or lower bank) of the appropriate data module (i.e., upper or lower buffer store).

It will be noted that Table I and the Exhibits I to V refer to the various portions of buffer store and its organization in coded numerals and/or letters. The code is explained by reference to FIG. 4. Referring to FIG. 4 the upper module 304U of buffer store memory 304 is buffer module 1, whereas the lower module 304L is buffer module 2. The upper banks of buffer module 304U is row 1, or row upper whereas the lower bank of buffer module 304U is row 2 or row lower. Similarly, the upper bank of module 304L is row 1, or row upper and the lower bank is row 2 or row lower. Sixteen bytes are stored in a given column of a given row of a given module. Hence, a Hit 1 indicates a match has been made with a 32 byte word stored in buffer module 304U; whereas a Hit 1 upper indicates a match has been made with a 16 byte word stored in the upper bank (row upper) of upper module 304U (module 1).

It has been previously shown that data is stored in the buffer store in various modes. One mode is the 128 .times. 2 .times. 32 i.e., 128 columns each containing 1 block (32 bytes) of data; there being two buffer memory modules, each having 128 columns. Since each 16 bytes of each column forms a row, in a full block of 32 bytes there are two rows in a given column. It has previously been shown how to access any column and any 16 byte or 32 byte word in any of several modes. It was moreover shown that write channels have a maximum capability of writing a word 8 bytes wide. However, it is often necessary to write only a portion of a word 1 byte wide or 2 bytes wide up to 8 bytes wide. To do this it is necessary to develop mask fields 0 through 7 to mask out unwanted fields in order to write or read only portions of words. Referring therefore to those portions of FIGS. 8A, 8B, and 8C which are within the dash dot lines and are designated as .alpha. and also to Exhibit I, there is shown the logic block diagrams and logic expressions respectively for developing the initial conditions for replacing row 1 in buffer 1. Referring specifically to Exhibit I there is shown the logic expressions for generating a function (i.e., signal) BIWES (Buffer One Write Enable Set). Exhibit II shows the logic expressions or conditions for generating a function B2WES (Buffer Two Write Enable Set). These functions are similar and are similarly generated but pertain to different buffer modules. It will be noted from Exhibits I and II that there are eight paragraphs in each of the Exhibits and each paragraph sets out the condition for generating the B1WES or B2WES function depending on whether Exhibit I and Exhibit II is referred to. The conditions of each

statement represent the input signals of an AND gate and these AND gates are OR'ed together to feed an amplifier for generating the B1WES or B2WES signal.

To illustrate the foregoing refer to Exhibit I paragraph 1 which is a statement that says when validity bit one lower (V1L) and validity bit one upper (V1U) is a logic zero and when an activity bit (ACTB) is also logical zero and when an OK bit is a logical one then the function B1WES is generated. However, if V1L and V1U are both logical zero another signal Bv1SZ10 (Buffer Validity Bit 1, Set to Logical Zero) may be generated and this signal may be substituted for V1L and V1U equal logical zero. The result is shown in Exhibit I paragraph 1b. The significance of the activity bit equal to the logical zero is that it points to buffer 1 row 1 (upper bank of upper module); conversely if activity bit is equal to logical 1 it points to buffer 2 row 2 (lower bank of lower buffer module).

Paragraph 2b of Exhibit I makes the Statement that when the signal BV1SZ10 is logical zero and BV2SZ00 is logical zero not (meaning it is logical 1) and when OK bit is logical 1 then once again the signal B1WES is generated. Note that BV1SZ10 by the convention shown on FIG. 10 is in the affirmative state and could have been written BV1SZ i.e. Buffer Validity 1 Set to Zero; whereas BV1SZ00 is negative and could have been written BV1SZ i.e. Buffer validity 1 not set to zero. However, the notation shown are in this instance preferred in this disclosure because it is consistent with the convention previously described although the alternative expression is perfectly valid and will be sometimes used where it is easier to read. Once again it will be noted that the signal BV2SZ is generated by V2L and V2U which comes from a particular location in the buffer memory.

The third paragraph of Exhibit I makes the statement that if a 128 by 2 by 32 mode signal is true and that there has been a hit 1 stored (indicating the information desired is stored in the buffer store 1) and if the OK bit is true then once again the function B1WES is generated. The fourth paragraph of Exhibit I makes the statement that if the 128 by 2 by 32 byte mode is not true and a hit one lower is stored, and the OK bit is logical 1 then the signal B1WES is once again generated. The fifth paragraph states that if there is a bit 1 upper stored and the 128 .times. 2 .times. 32 mode is not present and OK bit is logical 1 then B1WES is generated. The sixth paragraph makes the statement that if V2L and V2U is not logical zero and activity bit is logical zero and the 128 by 2 by 32 mode is true and a hit 2 (hit in buffer memory module 2) is not stored and an OK bit is logical 1 then once again the function B1WES is generated. Paragraph 7 makes the statement that if V2L and V2U are not zero, and activity bit is zero, and the 128 by 2 by 32 mode is not true and there is not a hit 2 lower stored then the signal B1WES is generated. Finally paragraph 8 makes the statement that if V2L and V2U is not zero, and activity bit is logical zero, and the 128 by 2 by 32 mode is not true and hit 2 upper is not stored (i.e. no such hit occurred) and OK bit is logical 1 then the signal B1WES is again generated.

Exhibit II indicates the conditions to replace row 2 buffer 2. All theconditions of Exhibit II are identical to Exhibit I except that the conditions are reversed. For example in the first paragraph of each Exhibit I and II, instead of having BV1SZ10, which is the validity bit one lower and validity bit one upper equal to zero, BV2SZ10, which is validity bit 2 lower and validity 2 upper, equal to zero. Moreover where an activity bit is present in any of the paragraphs of Exhibit II they will be set to logical 1 instead of logical zero as in Exhibit I.

Exhibit III A and III B indicate the conditions for developing functions B1WM0 through B1WM7 and functions B2WM0 through B2WM0. (The B1WM 0 through 7 functions are the buffer one write mask 0 through 7 functions, and the B2WM 0 through 7 functions are the buffer 2 write mask 0 through 7 functions). The functions B1WES10 and B2WES10 previously developed are utilized in Exhibit III to develop the buffer work mask control signals. Paragraph 0 of Exhibit III A makes the statement that if function B1WES is true and the data write cycle (DWC) is true or if the signal B1WES is true and memory write cycle (MWC) is true and data write mask zero is true then a buffer one write mask control zero function (B1WM0) is generated. This generated signal points to the first byte of an 8 byte word which is to be masked. Similarly seven more functions are generated for bytes 1 through 7 of an 8 byte word associated with buffer memory 1. Exhibit III B indicates how the buffer 2 write mask control functions for an 8 byte word associated with buffer memory 2 are generated. Hence any number of bytes 0 to 7 in an 8 byte word may be masked i.e., not written into or read from buffer store.

Although the generation of only seven masking signals is herein typically described, other numbers lesser or greater may be utilized.

Once appropriate masking signals have been generated, they may be applied to the inhibit lines of an addressed word, to inhibit or mask that portion of the word in reading or writing. (See U.S. Pat. Nos. 3,223,984 and 3,110,017 for typical memories where such application can be made). See also pages 59-61 of Honeywell Computer Journal Winter, Spring 1968 for typical semiconductor memories, where such application may be made.

Referring to Exhibit IV there is shown the conditions of the various statements for developing the function BSV1U (Buffer Set Validity One Upper). Any one of the four statements will set the validity one upper bit i.e. validity bit for buffer memories 1, upper row. Statement number 1 says that the above statement is true when validity 1 lower and validity 1 upper is logical zero and validity 2 lower and validity 2 upper is not logical zero and the 16 byte load of the 128 by 2 by 32 mode is true and address bit 27 is set (i.e., is true). Statement 2 says that when buffer validity 1, (i.e. validity bit for buffer memory 1, includes rows upper and lower) stored equals zero is true and when buffer validity 2 (i.e. validity bit for buffer memory 2, upper and lower row) stored is equal to zero is not true and the buffer store is

operating on 32 byte mode, then once again the validity one upper signal is generated, i.e., validity one upper bit is set or equal to 1. The third statement says that buffer validity bit for buffer store 1 upper is set to logical 1 when buffer validity 1 set is zero (BV1SZ10) and activity bit is logical zero, and the buffer store is operating either in 128 by 2 by 16 mode or 256 by 2 by 16 mode. Statement number 4 states that the buffer validity one upper bit is set when buffer validity one stored is equal to logical zero and activity bit is logical zero and address bit 27 is set, and the buffer store memory is in the 128 by 2 by 32 mode.

Referring now to Exhibit V there is shown three statements which are utilized to develop the buffer validity one upper write function which in effect is the actual write mask for the validity bit one upper. The mask for the remaining validity bits may be developed in a similar manner. Statement 1 of Exhibit V says that write mask signal BV1UW is true when the function BSV1U10 is true and buffer directory write update cycle is true. This statement says that correct data is to be written into row 1 buffer 1 and therefore validity bit 1 upper must be set. Statement number 2 says that the write mask for buffer validity bit 1 upper, is developed when buffer directory write update cycle is true and buffer set validity one update is not true and the function buffer 1 validity upper set is true (the function B1VUS10 is developed in Exhibit VI). The third statement says that the write mask function for buffer validity bit 1 upper is developed by the I/O unit. The I/O unit invalidates validity bits only when new inputs from the I/O unit are made to the main memory and those inputs are also stored in the buffer store. Therefore the function BV1UW is true when the function BIH1U10 (buffer I/O hit on buffer store one upper has been made) and a function BIUDC30 is true indicating that there is a buffer I/O update cycle.

Referring now to Exhibit VI there are shown 6 statements for the function B1VUS to be true. The first statement says that the statement B1VUS is true when activity bit is logical zero and the buffer store is either in the 128 by 2 by 16 or the 256 by 2 by 16 modes. (861610) The second statement says that the function B1VUS is true when there has been a hit one lower and is stored (BH1LS10) and the buffer store is operating in the 128 by 2 by 32 mode with a 16 byte load (B823230). Statement 3 states that BV1US is true when activity bit is set to logical zero and the mode is 128 by 2 by 32 and there is no hit 2 lower (BH2LS) stored. The fourth statement says that function B1VUS is true when activity bit is logical zero and there has been no hit in buffer 2 upper BH2US00 and the unit is operating in 128 by 2 by 32 byte mode with a 16 byte load. The fifth statement says that B2VUS is true when the activity bit is a logical zero and there has been no hit in buffer 2 (BH2S00) and the unit is operating in a 32 byte load mode.

Having developed the various functions (i.e., signals) for implementing the invention the logic circuitry which will generate equivalent signals normally follows. The logic circuitry for developing the signal B1WES10 is shown on FIG. 8C labeled .alpha. and enclosed by dash-dot lines, and includes AND gates 892C-898C and 8010C-8013C and amplifier 899C; the logic circuitry for generating signal B2WES10 is shown on FIG. 8A and includes AND gates 801A-806A and 808A-812A and also amplifier 807A. The logic circuitry for generating signals B1WM000-B1WM700 and B2WM000-B2WM700, a total of 16 signals is shown on FIG. 8A in that portion of FIG. 8B labeled .alpha. down to and including AND gate 830B and amplifier 827B. The logic circuitry for developing signals BSV1L10 through BSV1U10 and signals BSV2L10 through BSV2U10 is similar to each other and is shown on FIG. 8B within the dash-dot lines labeled B beginning with AND gate 831B down through and including AND gate 863B and all amplifiers related thereto. Similarly, the logic circuitry for generating signals B1VLS10 through B1VUS00 are shown on FIG. 8B also, beginning with, and including all AND gates and amplifiers from AND gate 863B to AND gate 882B. Signals B2VLS through B2VUS may be generated with similar logic circuitry. Also, the logic circuitry for generating signals BV1LW00 through BV2LW00 and signals BV1UW00 to BV2UW00 are physically and functionally similar to each other and are shown on FIG. 8B as that circuitry comprising AND gates and amplifier starting with AND gate 883B down to and including AND gate 897B. FIG. 8C shows logic circuits which store the signals BH1LS00 and BH1LS10, BH1US00 and BH1US10, BH2LS00 and BH2LS10, BH2US00 and BH2US10, BH2ST 00 and BH2ST10, BOKBS00 and BOKBS10, BACTS00 and BACTS10, BV1LS00 and BV1LS10, BV1 US00 and BV1US10, BV2LS00 and BV2 LS10, and BV2US00 and BV2US10. Utilizing the convention for the signal names and the symbols shown in the prior art diagram FIG. 10, Exhibits I through VI and Table I the FIGS. 8A through 8D are self-explanatory.

For example, to generate the signal B2WES10 on FIG. 8A, it is only necessary to OR together AND gates 801A-805A and apply the output of these gates as one input of AND gate 806A. The other input terminal of AND gate 806A has the signal BOKBS10 applied to it. Also AND gates 809A-811A are also OR'ed together with their output applied as one input of AND gate 808A. The other input signals applied to input terminals of AND gate 808A are BV1SZ00, BACTS10 and BOKBS10. AND gates 806A and 808A are then OR'ed together and their output signal is applied to the input terminal of amplifier 807A which generates the desired signal B2WES10. Following through FIGS. 8A8D they become self-explanatory when taken in conjunction with the convention hereinbefore defined.

Referring now to FIG. 9A, there is shown timing diagrams for a CP read with no hit and a CP read with a hit. The CPGO signal is a cycle generated within the CP that informs the buffer that a cycle is requested by the CP. The IOCGO signal is a comparable cycle that informs the buffer store that the I/O unit requests a cycle. Where there is contention for the buffer store between the CP and the IOC, the buffer store memory is allocated to the CP first. The BCPDC10 signal is a CP directory cycle; it is during this cycle that there is a determination as to whether or not the address sent by the CP is within the buffer memory and therefore has a "hit" or not. If no hit is obtained during this cycle, the function 10 (ninth cycle from the top) is set. With a no hit in the buffer store directory, main memory is accessed to obtain the data requested by the CP. The buffer store system 300 on FIG. 3 then initiates two cycles BM1PF10 and BPBCB10. During the BM1PF10 cycle, the buffer store system 300 accesses the first eight bytes of

data from main memory and sends four of the eight bytes to the CP and retains eight bytes. The BPBCB10 signal is a buffer busy signal and prevents any successive CP requests from entering the buffer store during this cycle. This signal will stay high until the four main memory requests from the CPU are fulfilled. Referring now to the fourth signal from the top BIGOS10 is utilized by the priority resolution logic to resolve any pending IO directory cycle conflicts. The BIODC10 signal is the I/O directory cycle which permits the I/O unit 307 to check the buffer directory module 305 for a hit. The illustrated case here indicates that the I/O unit did not make a hit and therefore releases the buffer directory. Had there been a hit, the signal B1UDC10 (Buffer 1 Update Cycle) would have gone high so that the I/O unit could have updated the buffer 1 store directory module. However, since in this instance there was no hit, the buffer IO update cycle is low and no update is made on the buffer directory. The CPGO reset signal is the inverse of the CPGO signal and acknowledges to the CP that it can reset the GO signal. The signal BNMGO10 is the GO signal issued by the buffer to the main memory to indicate that there has been a CP no hit condition in the buffer unit 300 and the buffer unit is issuing a GO request signal to the main memory in order to obtain the required information. The Buffer GO Reset signal below that is a cycle used by the main storage sequencer (See application Ser. No. 295,331 filed concurrently with the instant application) to acknowledge receipt of the buffer's GO signal and to indicate to the buffer to reset its GO signal. The NBACK10 signal is an acknowledge signal from main memory to the buffer indicating to the buffer that the main memory is handling the buffer request and the buffer can generate a new address or request. The READ STROBE signal is a signal from the buffer unit to the CP informing the CP that the four bytes requested by it are being delivered. The BMSCF1.phi. signal starts the counters used to count the number of clock cycles from the memory acknowledge signal to a time when the data is valid in the Buffer Store unit. It is utilized to determine whether or not there have been any timing slips on the buffer to main memory interface. The BMAC cycles 1-6 are counting cycles utilized to determine whether there have been any slips or not. The DUMMY HIT cycle is utilized only during the CP first pass request and it is utilized to install the CP if it has stopped writing for memory serivce. When the CP makes a request to the buffer, its clock is conditionally shut off and it is stalled and the DUMMY HIT signal starts the clock up again. The BMH1F10 signal is an error indication cycle. The BMDWC10 cycle is the data write cycle and is the interval during which data from main memory is written into the buffer modules. The BWCC1 and BWCC2 cycles are utilized to count the number of main memory requests that have been made by the buffer. The BDWUC10 signal is the data update cycle and is high only if an error has occurred during the four main memory accesses. If an error has occurred, the buffer unit zeroes the validity bits to indicate that the data contained in the buffer is not valid because of an error that occurred during writing. The BDWC1 signal is a cycle utilized to increment the address bits. The BDWUB signal is a buffer write update busy signal which solves conflicts between the I/O unit and the buffer. It prevents access by the I/O unit to the buffer during this period. The functions below this BNA27-28, BMA27-28, and BSA27-28 are address bits for incrementing the address for accesses to the different modules of main memory.

Having shown and described one embodiment of the invention, those skilled in the art will realize that many variations and modifications can be made to produce the described invention and still be within the spirit and scope of the claimed invention.

SPECIFICATION EXHIBITS

EXHIBIT I

Conditions for developing the function b1wes

1. (a) Validity bit 1 lower (V1L) and validity bit 1 upper (V1U) is logical zero AND activity bit (ACT B) is logical zero AND OK bit is logical 1; however, when V1L and V1U are both logical zero signal BV1SZ10 (Buffer Validity Bit 1 set to logical zero) is generated, then (1a) is:

(b) BV1SZ10 is logical 0. ACT B is logical zero. OK Bit is logical 1

+

2. (a) Validity bit 2 lower (V21) and validity bit 2 upper (V2U) is logical zero and V1U is logical zero AND-OK bit is logical 1 since when validity bit 2 lower (V2L and validity bit 2 upper (V2U) is logical zero signal BV2SZ (Buffer validity Bit 2 set to logical zero) is generated, the (2a) is:

(b) BV1SZ10 is logical 0 @. BV2SZ00 is not logical zero @. OK Bit is logical 1

+

3. 128 .times. 2 .times. 32 Mode @. Hit 1 is stored @. OK bit is logical 1
4. 128 .times. 2 .times. =Mode @. Hit 1 lower is stored @. OK bit is logical 1

+

5. 128 .times. 2 .times. 32 Mode @. Hit 1 upper stored @. OK bit is logical 1

+

6. V2L. V2U is zero @. ACT B is zero @. 128 .times. 2 .times. 32 Mode . HIT 2 STORED @. OK bit is

+
7. V2L. V2U is zero @. ACT B is zero @. 128 .times. 2 .times. 32 Mod@ HIT 2 lower stored

+
8. V2L. V2U is zero @. ACT B is zero @. 128 .times. 2 .times. 32 Mode@. Hit 2 upper stored@. OK bit is logical 1

=

B1WES10

EXHIBIT II

Conditions for developing the function b2wes
1. (a) (V2L. V2U is logical zero). ACT B is logical 1. OK bit is logical 1

or
1. (b) BV2SZ10. ACT B is logical. OK bit is logical 1

+
2. (a) (V2L. V2U is logical zero). (V1L.V. 1U is logical zero). OK bit is logical 1

or
2. (b) BV2SZ10 @. BV1SZ00 @. OK bit is logical 1

+
3. (a) 128 .times. 2 .times. 32 Mode. Hit 2 is stored. OK bit is logical 1. Since HIT 2 STORED IS BH2ST @. OK bit logical 1
(b) 128 .times. 2 .times. 32 Mode @. BH2ST @. OK bit logical 1

+
4. 128 .times. 2 .times. 32. HIT 2L STORED @. OK bit is logical 1

+
5. 128 .times. 2 .times. 32 @. HIT 2U STORED @. OK bit is logical 1

+
6. (V1L. V1U is zero) @. ACT B is logical 1. 128 .times. 2 .times. 32 Mode. HIT 1 STORED. OK bit is logical 1
7. (V1L. V1U is zero) @. ACT B is logical 1 128 .times. 2 .times. 32 Mode @. HIT 1 LOWER STORED OK bit to logical 1

+
8. (V1L. V1U is zero) @. ACT B is logical 1 128 .times. 2 .times. 32 Mode @. HIT 1 UPPER STORED OK bit is logical 1

=

B2WES10

EXHIBIT III A

Conditions for developing write mask bits b1wm0 through B1WM7
0 (b1wes10 @. dwc30) + (b1wes10 @. mwc@. dwm0030) = b1wm000
1 (b1wes10 @. dwc30) + (b1wes10 @. mwc@. dwm0130) = b1wm100
2 (b1wes10 @. dwc30) + (b1wes10 @. mwc@. dwm0230) = b1wm200
3 (b1wes10 @. dwc30) + (b1wes10 @. mwc@. dwm0330) = b1wm300
4 (bwes10 @. dwc30) + (b1wes10 @. mwc@. dwm0430) = b1wm400
5 (b1wes10 @. dwc30) + (b1wes10 @. mwc@. dwm0530) = b1wm500
6 (b1wes10 @. dwc30) + (b1wes10 @. mwc@. dwm0630) = b1wm600
7 (b1wes10 @. dwc30) + (b1wes10 @. dwm0730) = b1wm700

exhibit iii b

conditions for developing write mask bits b2wm0 through B2WM7
0 (b2wes10 @. dwc30) + (b2wes10 @. mwc@. dwm0030) = b2wm000
1 (b2wes10 @. dwc30) + (b2wes10 @. mwc@. dwm0130) = b2wm100
2 (b2wes10 @. dwc30) + (b2wes10 @. mwc@. dwm0230) = b2wm200
3 (b2wes10@. dwc30) + (b2wes10 @. msc@. dwm0330) = b2wm300

4 (b2wes10 @. dwc30) + (b2wes10 @. mwc@. dwm0430) = b2wm400
5 (b2wes10 @. dwc30) + (b2wes10 @. mwc@. dwm0530) = b2wm500
6 (b2wes10 @. dwc30) + (b2wes10 @. mwc@. dwm0630) = b2wm600
7 (b2wes10 @. dwc30) + (b2wes10 @. mwc@. dwm0730) = b2wm700

exhibit iv

conditions for developing the function (buffer set validity 1 upper)
1. a) (V1L@. V1U) is logical zero @. (V2l@. V2U) is logical zero @. 128 .times. 2 .times. 32 mode 16 byte load @. Address bit 27

or
(b) BV1SZ10.BV2SZ00 @. 128 .times. 2 .times. 32 Mode @. Address bit 27

+
2. BV1SZ10 @. BV2SZ00 @. 32 byte mode

+
3. BV1SZ10 @. ACT B is logical zero @. (128 .times. 2 .times. 16 mode + 256 .times. 2 .times. 16 mode)

+
4. BV1SZ10 . ACT B is logical zero @. 128 .times. 2 .times. 32 mode @. Address bit 27

=

BSV1U10

EXHIBIT V

Conditions for developing function bv1uw (buffer validity 1 upper write)
1. bsv1u10 @. bdwuc30
+
2. bdwuc30 @. bsv1vu00 @. b1vus10
+
3. bih1u10 @. biudc30
=
bv1uw

exhibit vi

conditions for developing the function b1vus (buffer 1 validity upper set)
1. b861610@. act b is logical zero

+
2. (Hit 1 lower stored.) @. B823230 (16 byte load)

+
3. BACTS is logical 0 @. B823230@. HIT 2 Lower Stored

+
4. ACTS is logical zero @. B823230 HIT 2 Upper Stored

+
5. B32BM30 @. HIT 2 STORED @. ACTS is logical 0

=

B1VUS10

SPECIFICATION TABLE

TABLE I

SIGNAL/FUNCTION DEFINITIONS
Sig./Funct. Name Definition

1. BV1SZ00 V1L and V1U stored are not zero.

2. BV1SZ10 V1L and V1U stored are zero.
3. V1L Validity bit, Buffer Store 1, Lower Bank.
4. V1U Validity bit, Buffer Store 1, Upper Bank.
5. V2U Validity bit, Buffer Store 2, Upper Bank.
6. V2L Validity bit, Buffer Store 2, Lower Bank.
7. BV2SZ00 V2L and V2U stored are not zero.
8. BV2SZ10 V2L and V2U stored are zero.
9. BACTS10 Buffer Activity Bit Stored in a flip-flop
FIG. 8C.
10. BACTS00 Activity Bit Not Stored in flip-flop.
11. BV2SZ 00 10 V2L and V2U stored are/are not zero,
depending on suffix: 00=No; 10=Yes.
(Suffixes will hence forth be left off, in
this table.)
12. BH1ST CP hit on Buffer-Store 1 Stored; Yes/No.
13. BH2ST CP hit on Buffer-Store 2 Stored; Yes/No.
14. BH1LS CP hit on Buffer-Store 1 Lower Bank,
Stored; Yes/No.
15. BH1US CP hit on Buffer-Store 1, Upper Bank,
Stored; Yes/No.
16. BH2LS CP hit on Buffer-Store 2, Lower Bank,
Stored; Yes/No.
17. BH2US CP hit on Buffer-Store 2, Upper Bank,
Stored; Yes/No.
18. BIHTL 18-23 are same as 12-17 except I/O unit
makes "hit."
19. BIHTU
20. BIH1L
21. BIH1U
22. BIH2L
23. BIH2U
24. BPSTE Stored Error.
25. DIAGM Diagnostic Mode
20. DIMWC Diagnostic Mode Write Cycle
21. BPDHE Diagnostic Mode Error
22. MPSWL Maint. Panel Switch
23. BPMWC Data Module Write Cycle
24. DWMO CP Write Mask, Byte 0
25. DWM1 CP Write Mask, Byte 1
26. DWM2 CP Write Mask, Byte 2
27. DWM3 CP Write Mask, Byte 3
28. DWM4 CP Write Mask, Byte 4
29. DWM5 CP Write Mask, Byte 5
30. DWM6 CP Write Mask, Byte 6
31. DWM7 CP Write Mask, Byte 7
(32-
40) B1WMO-7 Buffer-Store 1 Write Control Bytes 0-7
(40-
48) B2WMO-7 Buffer-Store 2 Write Control Bytes 0-7
49. BAB 27 Buffer Address Bit 27
50. BDWUC Buffer Directory Write Update Cycle
51. BIUDC Buffer I/O Update Cycle
52. BSV1U Buffer Set Validity Bit 1 Upper
53. BSV2U Buffer Set Validity Bit 2 Upper
54. BSV1L Buffer Set Validity Bit 1 Lower
55. BSV2U Buffer Set Validity Bit 2 Lower
56. B1WES Buffer-Store 1 Write Enable Set
57. B2WES Buffer-Store 2 Write Enable Set
58. BV1UW Buffer Validity Bit 1 Upper Write
59. BV1LW Buffer Validity Bit 1 Lower Write
60. BV2UW Buffer Validity Bit 2 Upper Write
61. BV2LW Buffer Validity Bit 2 Lower Write
62. BV1LS Buffer Validity Bit 1 Lower Stored
63. BV1US Buffer Validity Bit 1 Upper Stored
64. BV2LS Buffer Validity Bit 2 Lower Stored
65. BV2US Buffer Validity Bit 2 Upper Stored
66. BCPDC Buffer CP Directory Cycle

67. BCBCB Buffer Cycle Busy
68. BOKBS Buffer OK-Bit Stored
69. BDWUC Buffer Directory Write Update Cycle
70. CPDAT CP Data
71. BPAWC Buffer Processor Activity Write Cycle
72. B32BM Buffer 32 Byte Mode
73. B8232 128.times.2.times.32 Mode
74. B8616 128.times.2.times.16 or 256.times.2.times.16 Mode
75. BOKWE BOK Write Enable
76. BACTB Buffer Activity Bit
77. BDV1L Buffer Directory Validity 1 Lower
78. BDV1U Buffer Directory Validity 1 Upper
79. BDV2L Buffer Directory Validity 2 Lower
80. BDV2U Buffer Directory Validity 2 Upper
81. BPAWC Buffer Processor Activity Write Cycle
82. BCPDC Buffer CP Directory Cycle
83. BTMWC Memory Write Cycle
84. BPWDE Processor Data Error
85. UBWAB Processor Write Abort
86. BPAPE Processor Parity Error
87. BIDHE I/O Dual Hit Error – 2 Hits at same time
88. BIOWA I/O Write Abort
89. BPDHE Processor Dual Hit Error
90. BIODC I/O Directory Cycle
91. BPBCB Buffer Processor Cycle Busy
92. BLOG1 Logical 1 (Grounded Wires)

Claims

1.
1. A variable masking apparatus for a computer system, having a storage device comprising n modules of storage with each storage module having inhibit means for inhibiting information from being written into or read out of said storage module, said variable masking apparatus for providing masking signals for simultaneously masking under program control selected ones of a variable number of a sub-group of b bytes of information comprising a portion of a selected group of i bytes which may be stored in predetermined locations in any of said n modules of the storage device that is capable of operating in any of m storage modes comprising a normal mode and at least two reconfigured modes, said apparatus comprising: a. first means coupled to said n modules for addressing a selected one of any of said n modules of the storage device when said storage device is operating in a selected one of any of said m modes; b. second means coupled to said first means and to said n modules for addressing a selected group of i bytes of information in the selected one of said n modules; and, c. variable masking signal means, coupled to said second means and to said n modules, for applying masking signals to said inhibit means, said masking signals indicating selected ones of said b bytes of said selected

2. The variable masking apparatus as recited in claim 1 werein said storage device may have B modules, each module having A columns capable of storing a block of C bytes and wherein said modes of operation of said storage device include a normal mode (A by B by C) wherein said storage device is capable of storing (A) columns of one block (i.e. C bytes) or a half-block (i.e. C/2 bytes) of information per column; and (A by B by C/2) mode wherein said storage device is capable of storing A columns of a half-block of information per column; and an (E by B by C/2) mode wherein said storage device is capable of storing E columns of half-block of

3. The variable masking apparatus as recited in claim 2 wherein said storage device is comprised of a main memory which is k-way interleaved operable in one mode and a buffer store of smaller capacity and faster access time than said main memory operable in any of m modes under program

4. The variable masking apparatus as recited in claim 3 wherein said modes of operation of said buffer store include a by-pass mode wherein said buffer store is not utilized and all accesses for information are made to main memory, and including means coupled to said main memory and responsive to a predetermined instruction in a program then under

5. In combination with a general purpose computer system having a multi-level storage system, a variable masking apparatus comprising: a. a main memory for storing blocks of information each block comprised of C bytes; b. a buffer store comprising at least two modules, each module for storing information in a selected one of a plurality of multiple-length-byte word modes and each module having inhibit means for inhibiting portions of information of said multiple-length-byte word from being written into or read out of said module; c. first means coupled to said buffer store and responsive to a first predetermined instruction of a program then being under execution by said computer system, said first means for dynamically altering the existing storage mode of operation of said buffer store; d. directory means,

coupled to said buffer store, said directory means for storing addresses of said main memory wherein the information indicated by the addresses of said main memory is also stored in said buffer store as well as in said main memory; e. second means coupled to said main memory, said directory means and said buffer store, for comparing information in said directory with information in a selected instruction of said program then being under execution, whereby it is determined whether or not information requested by the computer system is in said buffer store; f. third means coupled to said main memory and buffer store, for addressing a selected one of any of the modules of said buffer store when said buffer store is operating in a selected one of said multiple-length byte modes; g. fourth means coupled to said main memory, buffer store and third means, for addressing a selected word in said addressed module; and, h. variable masking means coupled to said buffer store and to said third and fourth means, for providing signals to said inhibit means for masking

6. The combination as recited in Claim 5 wherein said buffer store comprises B modules, each module having A columns capable of storing a block of C bytes and wherein said modes of operation of said buffer store include a normal mode (A by B by C) wherein said buffer store is capable of storing A columns of one block (i.e., C bytes) or a half-block (i.e. C/2 bytes) of information per column; and (A by B by C/2) mode wherein said buffer store is capable of storing A columns of a half-block (i.e. C/2 bytes) of information per column; and (E by B by C/2) mode wherein said buffer store is capable of storing E columns of half-block of information (i.e. C/2 bytes) per column, and wherein E is greater than A.

7. The combination as recited in claim 6 wherein said modes of operation of said buffer include a by-pass mode wherein said buffer store is not

8. The combination as recited in claim 7 wherein main memory is k-way

9. The combination as recited in claim 8 including means, coupled to said main memory and responsive to an instruction of a a program then under

10. A variable masking apparatus for a computer system, said variable masking apparatus for simultaneously accessing, under program control, a variable number of bytes b of information stored in groups of i bytes per group in any of N modules of a storage device that is capable of operating in any of m storage modes each of said modules of said storage device having inhibit means for inhibiting information from being written into or read out of said storage, said masking means comprising: a. first means, coupled to each of said N modules of said storage device, responsive to signals indicative of the storage mode and state of said N modules for generating buffer-store N write enable set signals (BNWES) indicative of the storage module N enabled for writing information into said N storage module; b. second and third and fourth means, coupled to said N modules of said storage device, for generating a data write cycle signal (DWC), a memory write cycle signal (MWC), and a data write mask signal (DWMXX); and, c. fifth means, coupled to said first, second, third, and fourth means, said fifth means responsive to signals representative of the following Boolean expression; (BNWES@. DWC) + (BNWES@. MWC@. DWMXX) said fifth means for generating write mask control signals, (BNWMY), predetermined ones of said write mask control signals (BNWMY) being associated with predetermined bytes of said group of i bytes where Y indicates the byte of said group of i bytes with which the signal BNWMY is associated, said write mask control signal BNWMY for indicating, when said write mask control signal is low, the number Y of the byte of said group

11. In combination with a general purpose computer system having amulti-level storage system, a variable masking apparatus comprising: a. a main memory for storing blocks of information, each block comprised of C bytes; b. a buffer store comprising at least two modules identified as number 1 and number 2 respectively, with each of said modules being divided into an upper and lower module for storing information in a selected one of a plurality of multiple-length-byte word modes, and with each of said upper and lower modules having inhibit means for inhibiting information from being written or read out of said modules; c. first means, coupled to said buffer store and responsive to a first predetermined instruction of a computer program then being under execution by said computer system, said first means for dynamically altering the existing storage mode of operation of said buffer store; d. directory means, coupled to said buffer store, said directory means for storing addresses of said main memory is also stored in said buffer store as well as in said main memory; e. second means coupled to said main memory, said directory means and said buffer store, for comparing in-formation in said directory with information in a selected instruction of said program then being under execution, whereby it is determined whether or not information requested by the computer system is in said buffer store; f. third means, coupled to said main memory and buffer store, for addressing a selected one of any of the modules of said buffer store when said buffer store is operating in a selected one of said multiple-length-byte modes; g. fourth means coupled to said main memory, buffer store and third means, for addressing a selected word in said addressed module; h. fifth means, coupled to said third means, for generating validity signals validity-1-upper (V1U), validity-2-upper (V2U), validity-1-lower (V1L) and validity-2-lower (V2L) signals, said (V1U) signal for indicating when high, that data written in upper module 1 is valid, said V2U signal indicating, when high, that data written in upper module 2 is valid, said (V1L) signal for indicating, when high, that data written in the lower module 1 is valid, said (V2L) signal for indicating, when high, that data written in the lower module 2 is valid; and, i. variable masking means coupled to said buffer store and to said third, fourth and fifth means for generating inhibit signals for application to

12. The combination as recited in claim 11 including, buffer count means and error store means coupled to said buffer store and to said third, fourth and fifth means respectively for generating and storing error

13. The combination as recited in claim 11 including acitivity field means, coupled to said buffer store and to said third, fourth and fifth means, for indicating the least recently used upper or lower rows in said buffer

14. The combination as recited in claim 13 including O.K. means, coupled to said buffer store and to said third, fourth and fifth means, and associated with a predetermined address word in said directory means, said O.K. means for indicating that the address word associated with it has not been invalidated.
